Prediction Assignment

Joy SN

January 6, 2019

Build a predictive model for the manner in which subjects did the exercise.

Promblem Summary

To analyse the data colected using FuelBand, Fitbit (accelerometers) about subject's movement during different types of excercise. We need to build a model which can predict the type of "barball lifts"" ("classe") based on different movement patterns recorded by the subjects.

Data

The training data for this project are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:
https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Processing

```
## [1] 19622   160
```

Data contains 160 columns and 19622 rows. So we have ideally 160-1 predictors.
We convert 2 columns with factors to number format - user_name, new_window

With head() function we saw many columns have NA. So we are removing all columns which has more than 10% NAs We have decided not to impute any Imputing data

```
## [1] 19622    58
```

Now we have 57 columns to chose the list of predictors from.

We have 3 columns related to time, raw_timestamp_part_1, raw_timestamp_part_2, cvtd_timestamp and

```
## [1] 19622    54
```

After this we have 54 columns to chose the list of predictors from.

Exploratory data analyses

We divide our test set into testSet and ValidationSet

What are some fields that have high correlations with the classe? Let us check if any of the columns are related to each other, if so, we will remove them

```
## [1] 0.1484384 0.1587872 0.2448565 0.2955696 0.3453585
```

Top 2 correlations are - magnet_arm_x, pitch_forearm

Using *Fig 1* We cannot see clearly any relationship with these 2 highly related columns. Probably we cannot select our columns by just seeing these columns correlation individually

Model selection

Let us try to identify variables with high correlations amongst each other in our set, so we can possibly exclude them from the pca or training.

Let us use PCA to reduce the number of columns

```
## [1] 14718    37
```

We see that using PCA we could reduce the nunber of principal components to

Now, let us generate some Random Forest training. We are using different number of trees 30,60,90,120 on testSet without PCA and testSet with PCA and checking the accuracy. This has been generated in *Table-1*

Model selection

```
## [1] "Accuracy on trainsEt(Without PCA) with Trees- 60 is: 0.981"

## [1] "Accuracy on testsEt(Without PCA) with Trees- 60 is: 0.982"

##
## Call:
##  randomForest(x = trainSet[, -classeIndex], y = trainSet$classe,
xtest = validationSet[, -classeIndex], ytest = validationSet$classe,
ntree = ntree, proximity = TRUE, keep.forest = TRUE)
##                Type of random forest: classification
##                      Number of trees: 60
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.35%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 4183    1    0    0    1 0.0004778973
## B    6 2836    5    1    0 0.0042134831
## C    0    6 2561    0    0 0.0023373588
## D    0    0   18 2393    1 0.0078772803
```

```
## E    0    1    1   10 2694 0.0044345898
##                 Test set error rate: 0.33%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1395   0   0   0   0 0.000000000
## B    2 946   1   0   0 0.003161222
## C    0   5 849   1   0 0.007017544
## D    0   0   4 800   0 0.004975124
## E    0   0   0   3 898 0.003329634
```

Based on the list of accuracy, we can say that PCA is not helping us much in this, except for reducing number of components. Also, with trees=60 gives us the best accuracy for test as well as train set. Increasing further is not increasing the accuracy. So we select a model without PCA and with number of trees = 60

Conclusion

This concludes that PCA is not helping us much in this, except for reducing number of components. So we will select model without PCA and with number of trees = 60 which has a test accuracy of 0.989 and train accuracy of 0.982 and OOB estimate of error rate: 0.34%

We draw some more plots in *Fig 2* for the final prediction to check chosen model

Test results

Although we've chosen the `rfFinal` it's still nice to see what the model with PCA would predict on the final test set. Let's look at predictions for all models on the final test set.

```
##         1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16
## cleaned "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E"
## pcaAll  "B" "A" "B" "A" "A" "E" "D" "B" "A" "A" "B" "C" "B" "A" "E" "E"
##         17  18  19  20
## cleaned "A" "B" "B" "B"
## pcaAll  "A" "B" "B" "B"
```

The predictions don't really change a lot with each model, but since we have most faith in the `rfFinal`, we'll keep that as final answer.

Appendix

Table 1: Accuracy of different Random Forest Models with and without PCA (Number of trees - 30,60,90 & 120)

```
##      No. tress Test(No PCA) Train(Non PCA) Test(PCA) Train(PCA)
## [1,]        30        0.970          0.984     0.758      0.862
## [2,]        60        0.979          0.991     0.846      0.887
## [3,]        90        0.985          0.980     0.859      0.896
## [4,]       120        0.983          0.984     0.880      0.890
```

Figure 1: Density plot of these columns with “classe” column

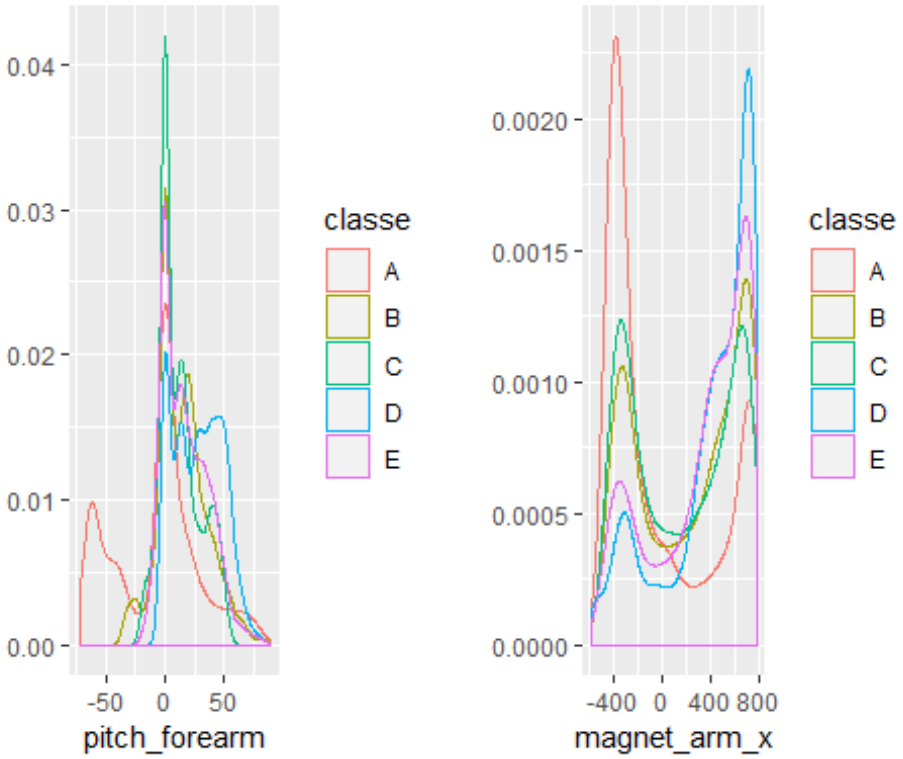

Figure 2: Variable Importance Plot for the Final Model and Error Vs No of Trees Plot